Serial and Parallel Implementations of Hybrid Fluid Model of Information Flows in Networks with Complex Topology

Dmitry Basavin[1*], and *Sergey* Porshnev[1]

[1]Ural Federal University named after first President of Russia B.N. Yeltsin, 620002 Yekaterinburg, Russia

Abstract. This paper is dedicated to the issue of modeling information flows in networks with complex topologies and it describes a comparison of the sequential (written in the MATLAB language) and parallel (based on GPGPU technology) software implementations of the hybrid fluid model (HFM) of Internet traffic. Obtained performance estimates of both software implementations indicate a higher performance of parallel software implementation HFM. The directions of further research, the results of which will be the basis for the later development of parallel software implementation HFM are proposed.

1 Introduction

In order to research features of the information flows, generated in modern computer networks with large number of different devices and applications, fluid model (FM) of network traffic [1] and its further modification – hybrid fluid model (HFM), which was proposed in [2], are used. FM and HFM represent the following coupled sets of nonlinear differential equations:

$$\frac{dW_i(t)}{dt} = \frac{\mathbf{1}_{(W_i(t)-M_i)}}{R_i(t)} - \frac{W_i(t)}{2}\cdot\lambda_i(t), \quad (1)$$

$$\frac{dq_l(t)}{dt} = -\mathbf{1}_{q(t)}\cdot C_l + \sum_{i\in N_l} A_i^l(t), \quad (2)$$

where $W_i(t)$ is TCP window size of a class i flow at time t, $\mathbf{1}_{f(t)}$ – Heaviside step function, $R_i(t)$ – round trip time of a class i flow at time t, $\lambda_i(t)$ – the loss indication rate experienced by a class i flow, $q_l(t)$ – queue length of a l link at time t, C_l – bandwidth of a link l, $A_i^l(t) = W_i^l(t)/R_i^l(t)$ – arrival rate of class i flow at link l at time t.

FM and HFM take into account the finiteness of data transmission time from one link to another (the delay), as well as feedback provided by *TCP* protocol, transmission rate control policies, active queue management algorithms as well as the interference of modeling flows.

Type of function $\lambda_i(t)$ is determined by links characteristics and traffic shaping algorithms, i.e. Rate limiting and AQM. From a physical point of view, these models are Bernoulli's fluid balance equations for corresponding links.

Due to lacking analytical solutions of ODEs, used in HFM, there is a need in finding their numerical solutions and consequentially developing appropriate software implementations. A version of HFM software implementation with sequentially executable code has been introduced in [3]. However, experience of its practical usage showed that the speed of data flows characteristics calculations is very slow even for relatively simple network configurations. In this regard, it was conjectured in [4] about the possibility of significantly increasing computational speed through the use of *GPGPU* technology. Authors have developed the parallel software implementation of *HFM* using *GPGPU* for networks with a single router in order to confirm it, and analysis of it was presented in [5].

The article touches upon the results of computational speed comparison of sequential and parallel software implementations of HFM for networks with multiple routers. Sequential implementation written with script programming language MATLAB [2] and parallel implementation written with C programming language and OpenCL framework were used in the experiments. Due to the fact of correspondence of computed numerical results obtained with sequential implementation of HFM and real network traffic was earlier proved in [2], we take convergence of sequential and parallel implementation as testifying criterion for correctness of parallel implementation.

2 The experiments methodology

As a part of experiments, it is suggested to carry out modeling of data flows in networks, which conceptual topology diagram is shown in Fig. 1.

Fig. 1 shows that network topology diagram of the described experiments, — is a collection of smaller networks, each of which is a subnet, consisting of two TCP-senders, one router, one receiver and three links. The total number of subnets for k-th experiment is $m = 2^{k-1}$, $k = 1, ..., 9$.

Each TCP class starts data transmission throughout the simulation time to maximize the algorithmic and computational hardware resources loading for performance evaluation and analysis of HFM software implementations. Variable HFM characteristics within a single experiment are as follows: the number of modeled TCP-senders, routers and receivers. Other HFM characteristics remain unchanged. This approach allows

*Corresponding author: basavind@gmail.com

us to conduct identical experiments on sequential and parallel implementations of HFM.

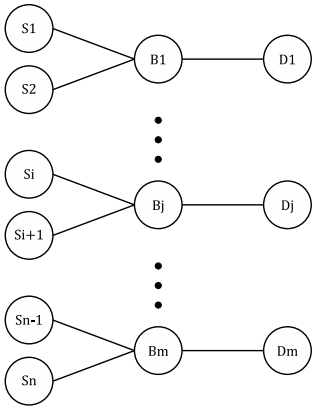

Fig. 1. Conceptual network topology diagram for experiment k, where S is TCP-sender, B – router and D – receiver

We are using next HFM characteristics in k-th experiment:

• Simulation time – 5 s,
• Number of TCP-classes – $n = 2^k$,
• Number of routers – $m = 2^{k-1}$,
• Average link propagation delay – 4 ms,
• Links bandwidth – 1 Mbps,
• Committed information rate (Rate Limiting algorithm) – 1 Mbps,
• Committed burst (Rate limiting algorithm) – 100 Kb,
• Excess burst (Rate limiting algorithm) – 1000 Kb.

Time measurements will be conducted before and after simulation, excluding initialization operations. We will compute 20 independent HFM solutions for experiment plan with obtained average computational time measurements over this ensemble of solutions. For each experiment all HFM solutions will be checked for consistency between sequential and parallel implementations.

The same hardware and operational system will be used for eliminating influence of external factors during the performance of each of the implementations during the experiments.

3 Analysis of experimental results

Comparison of the averaged over the ensemble of the independent solutions system of ODE (1)–(2), obtained using the sequential and parallel implementations HFM, showed that their differences from each other remained within the error due to the finite precision of the computer calculations.

Averaged over the ensemble of implementations of time simulation measurements for the 8 experiments plans, are described below in Fig. 2. As we can see from Fig. 2, computational time of sequential HFM implementation with any number of TCP-classes exceeds the same amount, and is measured in experiments with a parallel HFM implementation, indicating a higher performance of the latter. For a quantitative confirmation of this conclusion, Table 1 shows quantitative values of the computation time.

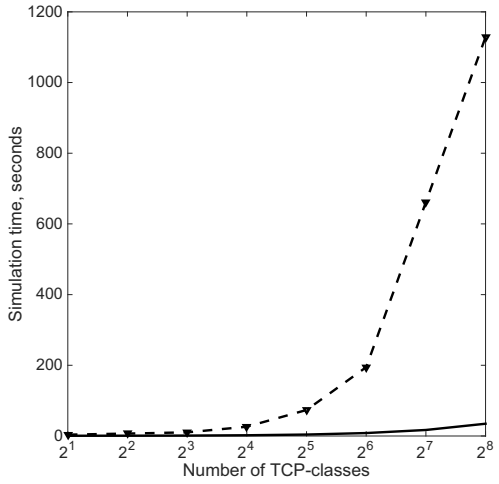

Fig. 2. The dependence of simulation time for the number of TCP classes: dotted line – sequential HFM implementation, solid line – parallel HFM implementation

Table 1. Comparative analysis of software implementations

Count of TCP-classes	Time of modeling in GPU, seconds	Time of modeling in MATLAB, seconds	Acceleration, time
2^1	0.684	3.760	5.494
2^2	0.818	6.823	8.338
2^3	1.186	10.245	8.635
2^4	2.084	26.224	12.583
2^5	4.083	73.505	18.005
2^6	8.329	194.542	23.357
2^7	16.960	661.531	39.005
2^8	34.699	1128.521	32.524

We should note, that, when using parallel processing algorithms on the central processor (CPU) (for example, embedded in the language MATLAB), the computational speed will decrease inversely proportional to the number of processor cores. However, even when using CPU with 8 cores computational time, it will obviously be more than when it is obtained with using a parallel implementation HFM. Analysis of the experience of using the sequential software HFM implementation, written with C language [5], shows that it provides higher speed of calculations in comparison with a software implementation, written with MATLAB script language, however, it will still be inferior to the parallel HFM implementation. But one of the main pros for using CPU implementations is the opportunity to use large amounts of RAM memory for storing parameters and calculation results, therefore, modeling a larger number of TCP classes of Internet traffic (according to our estimates, their number may reach 2^{17}) and networks with difficult topology.

As a part of a further improvement of the parallel software implementation, it is planned to develop a system architecture, which allows using dynamic-link modules in order to implement new algorithms AQM, TCP etc.

4 Conclusion

The efficiency of parallel HFM implementation, developed with usage of *GPGPU* technology, is confirmed. It is shown that its usage provides a significant increase in computation speed compared to the sequential HFM implementation, written in MATLAB language. Parallel HFM implementation, Created by the authors, in contrast to the packet simulators, i.e. NS-2, provides the possibility of modeling high speed Network with difficult topology.

References

1. V. Misra, W. Gong, and D. Towsley, Proceedings of ACM Sigcomm, 151 (2000).
2. M.K. Grebenkin, S.V. Porshnev, *Gibridnaia zhidkostnaia model magistralnogo Internet-kanala* (LAP LAMBERT Academic Publish., 2012) [In Rus]
3. M.K. Grebenkin, S.V. Porshnev, *Programmnaia realizatciia gibridnoi zhidkostnoi modeli informatcionnykh potokov v vysokoskorostnykh magistralnykh internet-kanalakh* (2012). [In Rus]
4. D.A. Basavin, S.V. Porshnev, *Proceedings of Aktualnye voprosy v nauchnoi rabote i obrazovatelnoi deiatelnosti*, 12 (2013) [In Rus]
5. D.A. Basavin, S.V. Porshnev, Proceedings of CriMiCo 2013 – 2013 23rd International Crimean Conference Microwave and Telecommunication Technology, 424 (2013).